Brenton Mac

DS210 Final Project Report

Objective

My objective for this project was to classify a network of online friends by how popular each person is, and find the celebrities within a set of people. To do this, I wanted to create a popularity scale, where the more popular people had a higher score. For my dataset, I used a twitch dataset from SNAP Stanford, named Twitch Social Networks: https://snap.stanford.edu/data/twitch-social-networks.html. Within this dataset, I used the Spanish one, which included 4648 vertices and 59383 edges. In order to classify each of the 4648 people, I calculated the node degree of each person. From there, I found the mean and standard deviation of the distribution. With these statistics, I was able to find the z-score of each node, where I could then classify each person by their z-score. With these scores, I found all the people with popularity scores of 7 (the highest score), and classified those people as celebrities.

How to Run

To run my code, cargo run can be used. My dataset is not large enough to require cargo run --release, however it can be used if the user wishes. The runtime for both cargo run and cargo run --release is 0.01 seconds. No errors are given when --release is used.

Methods

In order to implement my methods, I created 3 .rs files to split up my work: main.rs, stats.rs, and popular.rs.

For my stats.rs file, I created a struct called stats that has many functions implemented within the struct. All the struct takes in for parameters is the data, which is a vector of usize integers. The functions that are implemented are as follows:

- descriptive_stats: given the data, it returns the minimum, Q1, median, Q3, and maximum. Something to note within this function is how I calculated median and Q3 if the data is not divisible by 2 and 4 (would make median/Q1 singular values): I found the average between the 2 middle points, and the 2 points around the 75th percentile.
- mean: finds the mean of the data. I did this by adding all the values and dividing by the length of the data.
- stdev: finds the standard deviation of the data. I did this by taking the square root of the sum of the difference from the mean divided by the length of the data.
- zscores: finds the z score of each value based on the mean and stdev. I did this by finding the difference between the value and the mean, and then dividing by the standard deviation.

For my popular.rs file, I have two functions: popularity_scale, and celeb. popularity_scale gives each person a popularity score based on their z-score:

- Under -2.0: score of 1
- Between -1.0 and -2.0: score of 2
- Between 0.0 and -1.0: score of 3
- Between 1.0 and 0.0: score of 4
- Between 2.0 and 1.0: score of 5
- Between 3.0 and 2.0: score of 6
- Over 3.0: score of 7

These values are pushed to a vector and the vector is returned. For the celeb function, it iterates through the vector given by popularity_scale and returns all the node numbers that are popularity scores of 7, which are considered celebrities.

My main file main.rs, is used for aggregating all the functions created by the separate files, as well as defining the Graph struct. Within the Graph struct, there are several functions that are implemented into the struct: add_directed_edges, sort_graph_lists, create_directed, create_undirected, and node_degrees. Each of these has been imported from the lecture notes except node_degrees. This function calculates the node degree for each vertex. This is done by adding the amount of outgoing edges for each vertex and pushing assigning it to the same index as the vertex. Main also contains my read_csv function, which reads the csv file from a .csv into a vector of tuples, where the tuples are formatted (value one, value two).

When putting all these functions together in the main function, I first read the csv file that I have, “musae_ES_edges.csv”. I then found the max value within the csv file, which would be the amount of vertices there are. With this, I was able to create the graph struct, where I used create_undirected, and then found the node degree of each point. With the node degrees, I was able to transition to the stats struct, where I created the struct using the node degree data. With this data, I found the descriptive stats, mean, and standard deviation. These are printed. Then, I found the z-score, which then was used in the popular functions to find the popularity score of each person, and find the celebrities of the dataset.

Results

The output of my code is below:

```
There are 4648 vertices in the data.
The edges that are celebrities (7 on popularity scale) are [12, 68, 109, 182, 213, 291, 406, 499, 574, 596, 724, 735, 781, 811, 815, 847, 855, 896, 915, 940, 982, 1084, 1114, 1220,
 1254, 1264, 1279, 1351, 1376, 1381, 1397, 1486, 1565, 1585, 1687, 1715, 1728, 1800, 1819, 1822, 1826, 1890, 1919, 1971, 1976, 2018, 2112, 2278, 2283, 2294, 2306, 2388, 2408, 2475,
 2480, 2548, 2694, 2751, 2759, 2831, 2847, 2864, 2901, 2922, 3054, 3122, 3235, 3615, 3719, 3781, 3830, 3871, 4142, 4151, 4190, 4208, 4295, 4331, 4397, 4415, 4607].
There are 81 celebrities out of 4648 people
The mean of friendships every twitch user has is approximately 26.
The standard deviation for twitch user friendships is approximately 49.
Minimum: 1 | Q1: 5 | Median: 12 | Q3: 27 | Maximum: 1022
```

Something I noticed with this distribution is that the distribution does follow a power-law distribution. This means that large numbers are rare, and smaller numbers are much more common. This can be seen in the difference between the mean and median - the mean is 26, and the median is 12. This implies that the distribution is skewed right, which is the shape of a power-law distribution. Furthermore, the standard deviation is very high, implying that there is a lot of variance in the data. In this case, the variance comes in the shape of popular individuals who have many more friends than the normal person. This is what I expected, because social networks usually end up being similar to the power-law distribution. A Twitch dataset of friendships online is a perfect example of a social network, and it follows the trend of other social networks.

Something else interesting I noticed is that my data includes 81 celebrities, which is about 1.7% of the total data. This is interesting because my definition of a celebrity was if they had a z-score over 3, which should only be 0.15% of the data. This could further validate the idea of a power-law distribution because there are many celebrities that skew the distribution as they would in a power-law distribution.